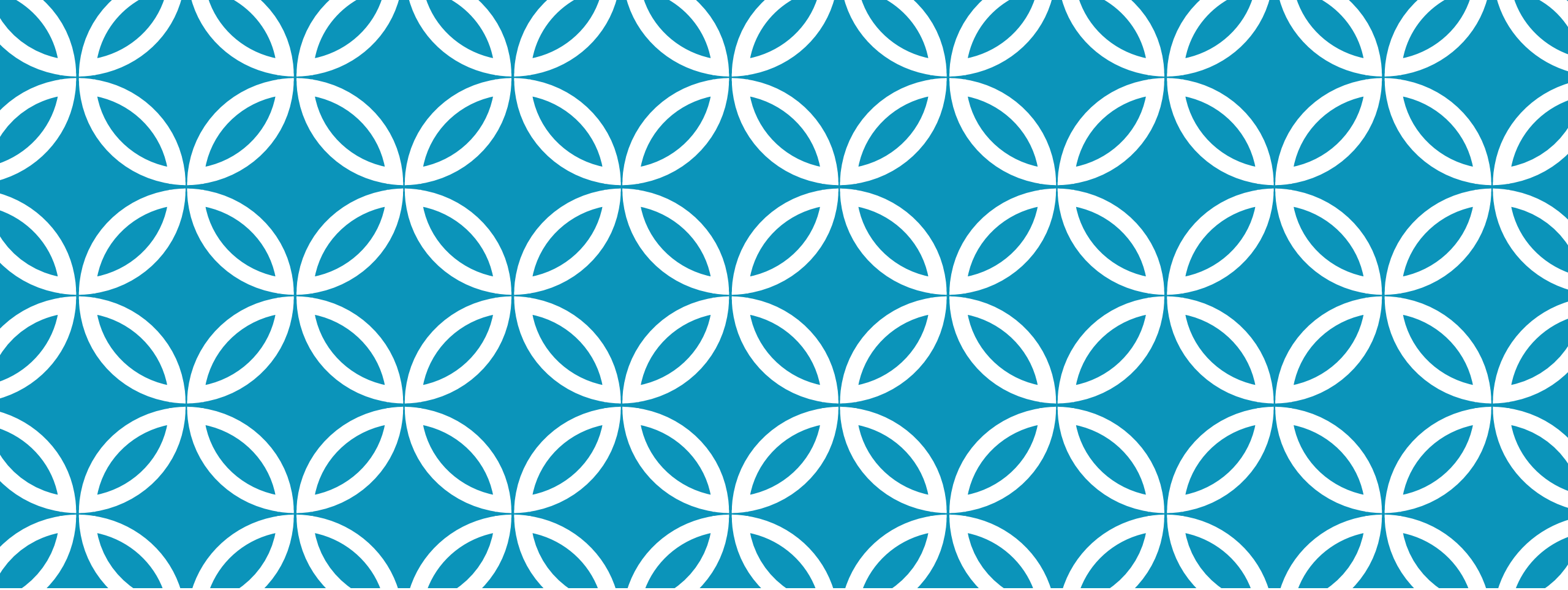


PYTHON FOR DATA SCIENCE

CS/EE/CNS/CMS/LOL/ROFL 155

Recitation 1

Kevin Tang

OUTLINE OF RECITATION

Python for Data Science

Installing Python, Numpy, Sk-learn

iPython notebooks

Basic Numpy

Linear Algebra “tricks”

More on expectation values

Additional Resources

CS 155 AND PYTHON

You may do sets in whatever (reasonable) language you are comfortable with

- Python and Matlab are recommended
- TA's may not be able to assist you with other languages

Python will be a supported language

- Solutions that require code will be released in either Python or Matlab
- Boilerplate/template code for projects and sets will be released in Python

We will support Python 2.7 and up

WHY PYTHON?

Free, Open source!

Everyone should know it from CS1 or have some familiarity

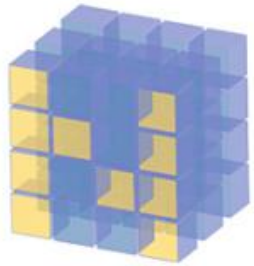
Relatively simple

Great packages: Numpy, Scipy, Sklearn, etc.

Not everyone has Matlab installed, and I know some who hate it with a passion

We will assume you know basic Python to at least a CS1 Level

NUMPY, SCIPY, MATPL



NumPy

Three packages in Python you will learn to know well

Numpy is a high-performance package that deals with arrays of data

- Matlab “like”
- Uses cython and bytecode to make things fast!

Scipy adds more data processing and other scientific tools

Matplotlib is a plotting library used to plot data

- Much prettier than Matlab plots

HOW TO INSTALL ALL OF THIS

Easy way: install Anaconda https://www.continuum.io/downloads

- Anaconda is a python *distribution* that has every tool you will ever use in this class and probably more
- After installing, go into command line/terminal and type

  "conda update –all" to update all of your packages.

Hard way: install Python from https://www.python.org/downloads/release/python-2711/

- Then install Scipy and Sklearn manually
- Alternatively, install miniconda rom continuum and use the cona command to manually install the package you need.

IPYTHON NOTEBOOK/ JUPYTER

Matlab/Mathematica like way to do your pythoning

Github supports rendering Jupyter notebooks! [Link]

Demo – see Ipython notebooks

MORE ON EXPECTED VALUES

Expected values are “averages” over some distribution

Expected values act **over** some domain

$E\downarrow D\,[f(x)]$ refers to the expected value of f(x) over data set D

$E\downarrow x\,[f(x)]$ refers to the expected value of f(x) over the entire domain of f

Properties of Expectation Values:

$E[\lambda X+Y]=\lambda E[X]+E[Y]$

$E[c]=c$

$E\downarrow D\,[E\downarrow x\,[f(x)]]=E\downarrow x\,[E\downarrow D\,[f(x)]]$

Question: is this true? $E[X\uparrow 2\,]=E[X]\uparrow 2$

EXPECTED VALUE OF DISCRETE RANDOM VARIABLES

Question 1: If you roll a die numbered 1-6, what is the expected value of the dice roll?

Question 2: If you roll a die numbered 1-6, what is the expected value of the square of the rolled number?

Question 3: If you are randomly drawing from a jar of numbers that contain the numbers [0, 1, 3, 4, 4, 6], what is the expected value of the number you draw?

Note that expectation value of discrete random variable X with some probability mass function $p(a)=P(X=a)$

$$E(X)=\sum ap(a)$$

In practice, you can simply take an average of all samples you have. The expectation value of some function f(x) over some dataset D with values $D\downarrow 1$, $D\downarrow 2$, $D\downarrow 3$,...can be calculated as

$$E(X)=1/|D|\sum i\in D\uparrow\blacksquare f(D\downarrow i)$$

EXPECTED VALUE OF CONTINUOUS RANDOM VARIABLE OVER A DOMAIN

Question 1: What is the expected value of a uniform distribution from [-1,1]?

Question 2: What is the expected value of $\cos(x)$ over the domain $[-\pi/2 ,\pi/2]$?

For some continuous domain [a,b] and some function f(x), the expected value of f(x)

$$E[f(x)]=1/b-a \int a\uparrow b\blacksquare f(x)dx$$

WHEN ANALYTICAL METHOD DOESN'T WORK

Say you have some unknown function f(x) (or its some unknown distribution that you can sample from) and you want to find the expectation value of g(f(x)) over a known domain D where g is a known function.

How can we do this?

Approximate!

Riemann Sums or Monte Carlo are potential methods here

For Monte Carlo, sample a large number of values randomly chosen from the distribution and average the g(f(x)) that you get!

ADDITIONAL RESOURCES

Python 2.7 documentation https://docs.python.org/2.7/

Scipy, numpy, matplotlib, pandas, and documentation http://www.scipy.org/

Scikitlearn: http://scikit-learn.org/stable/

Great Github repo of example python notebooks in data science: https://github.com/donnemartin/data-science-ipython-notebooks

SciPy Cookbook: collection of recipes for data science http://scipy-cookbook.readthedocs.org/

More in depth Python/Numpy tutorial http://cs231n.github.io/python-numpy-tutorial/